Learning with whom to communicate using relational reinforcement learning

Marc Ponsen, Tom Croonenborghs, Karl Tuyls, Jan Ramon, Kurt Driessens, Jaap van den Herik, and Eric Postma

Abstract Relational reinforcement learning is a promising direction within reinforcement learning research. It upgrades reinforcement learning techniques by using relational representations for states, actions, and learned value-functions or policies

Marc Ponsen
Department of Knowledge Engineering, Maastricht University,
Mindebroedersberg 6a, 6211 LK, Maastricht, The Netherlands
e-mail: m.ponsen@maastrichtuniversity.nl

Tom Croonenborghs
KH Kempen University College,
Kleinhoefstraat 4, 2440 Geel, Belgium
e-mail: tom.croonenborghs@khk.be

Karl Tuyls
Department of Knowledge Engineering, Maastricht University,
Mindebroedersberg 6a, 6211 LK, Maastricht, The Netherlands
e-mail: k.tuyls@maastrichtuniversity.nl

Jan Ramon
DTAI, Katholieke Universiteit Leuven,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
e-mail: jan.ramon@cs.kuleuven.be

Kurt Driessens
DTAI, Katholieke Universiteit Leuven,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
e-mail: kurt.driessens@cs.kuleuven.be

Jaap van den Herik
Tilburg centre for Creative Computing, Tilburg University
Warandelaan 2, PO Box 90153, 5000 LE Tilburg, The Netherlands
e-mail: h.j.vdnherik@uvt.nl

Eric Postma
Tilburg centre for Creative Computing, Tilburg University
Warandelaan 2, PO Box 90153, 5000 LE Tilburg, The Netherlands
e-mail: e.o.postma@uvt.nl

to allow natural representations and abstractions of complex tasks. Multi-agent systems are characterized by their relational structure and present a good example of a complex task. In this paper, we show how relational reinforcement learning could be a useful tool for learning in multi-agent systems. We study this approach in more detail on one important aspect of multi-agent systems, i.e., on learning a communication policy for cooperative systems (e.g., resource distribution). Communication between agents in realistic multi-agent systems can be assumed costly, limited and unreliable. We perform a number of experiments that highlight the conditions in which relational representations can be beneficial when taking the constraints mentioned above into account.

1 Introduction

Relational reinforcement learning (RRL) has emerged in the machine learning community as a promising subfield of reinforcement learning (RL) (e.g., [4,6,25,28]). It upgrades RL techniques by using relational representations for states, actions, and learned value-functions or policies to allow natural representations and abstractions of complex tasks. RRL offers a state space representation that is much richer than the one used in classical (or propositional) methods. This leads to a serious state space reduction, allowing the use of generalized prior knowledge and inferring general new knowledge as a result.

So far, most of the work in RRL has focused on single agent situations, i.e., environments in which only one agent operates and learns. Because multi-agent problems contain, besides a complex environment, also interactions between the agents, it is reasonable to assume that this new paradigm can also contribute to the multi-agent systems (MAS) domain.

An important source of complexity in MAS is the fact that agents only have partial information about the environment. Because of the inherent generalization and abstraction that comes with relational representations, agents can more naturally generalize over unseen parts of the environment. Also, in complex MAS, agents can interfere with each others actions, plans or goals. Hence, agents need to take these external influences into account and need to learn which other agents to keep in mind when optimizing ones own behavior. The latter can, for example, be achieved through communication . Through communication, agents can limit the influence of partial observability, and obtain valuable information about other agents. Harmful interactions with agents can be avoided, while on the opposite, agents with common interest can work together to achieve a higher reward. However, communication in complex MAS is not free. Communicating with other agents comes with a cost (e.g., the communication medium may not be free and/or have a limited bandwidth), is often limited (e.g., communication is only possible with agents within a certain range) or unreliable (e.g., messages may not always arrive error free or even not at all).

In the current paper we investigate the effect of using a relational representation with respect to agent communication. We define an abstract cooperative MAS, wherein agents can learn to communicate with each other to achieve a higher reward. More precisely, we consider a population of agents that each have to learn how to complete a number of tasks. The agents can have some prior knowledge about tasks, which allows them to complete the task more easily. Additionally, agents may ask other agents for advice. Hence, they can learn to complete a task not only by finding a solution themselves but also by asking an agent that already knows how to complete it. Therefore, it is important that agents have a good idea of who is good at which task, and as such obtain a good impression of the agent relationships. We perform a number of experiments that illustrate the benefit of using RRL for agent communication in MAS. More specifically, we empirically validate if (relational) communication is beneficial compared to no communication, and whether relational representations can cope with issues such as limited and unreliable communication. We also test whether our relational communication approach is scalable in terms of agents and task complexity. Our results illustrate that when using a relational communication approach, agents can learn to complete tasks faster through advice seeking.

The remainder of this paper is structured as follows: in Section 2 and 3 we explain RL and RRL respectively. The application of RRL in MAS, along with existing work, is discussed in Section 4. In Section 5 we describe our experiments that must empirically validate the advantages of using relational communication in MAS. We conclude in Section 6.

2 Reinforcement learning

Most RL research is based on the framework of Markov decision processes (MDP) [19]. MDP are sequential decision making problems for fully observable worlds. They are defined by a tuple (S,A,T,R). Starting in an initial state s_0 at each discrete time-step $t = 0,1,2,\ldots$ an adaptive agent observes an environment state s_t contained in a finite set of states $S = \{s_1,s_2,\ldots,s_n\}$, and executes an action a from a finite set of admissible actions $A = \{a_1,a_2,\ldots,a_m\}$. The agent receives an immediate reward $R : S \to \mathbb{R}$, that assigns a value or reward for being in that state, and moves to a new state s' depending on a probabilistic transition function $T : S \times A \times S \to [0,1]$. The probability for ending up in state s' after doing action a in state s is denoted as $T(s,a,s')$. For all actions a, and all states s and s', we have that $T(s,a,s') \geq 0$ and $\sum_{s' \in S} T(s,a,s') = 1$. An MDP respects the *Markov property*: the future dynamics, transitions and rewards fully depend on the current state, i.e., $T(s_{t+1}|s_t,a_t,s_{t-1},a_{t-1},\ldots) = T(s_{t+1}|s_t,a_t)$ and $R(s_t|s_{t-1},\ldots) = R(s_t)$. The transition function T and reward function R together are often referred to as the model of the environment. The learning task in MDP is to find a policy $\pi : S \to A$ for selecting actions with maximal expected (discounted) future reward. The quality of a policy is indicated by a *value function* V^π. The value $V^\pi(s)$ specifies the total amount of

reward which an agent may expect to accumulate over the future, starting from state s and following the policy π. Informally, the value function indicates the long-term desirability of states after taking into account the states that are likely to follow and the rewards available in those states. In a discounted infinite horizon MDP, the expected cumulative reward (i.e., the value function) is defined as:

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t) | s_0 = s\right] \tag{1}$$

A discount factor $\gamma \in [0,1\rangle$ is introduced to ensure that the rewards returned are bounded (finite) values. The variable γ determines the relevance of future rewards. Setting γ to 0 results in a *myopic* view on rewards (i.e., only the immediate reward is optimized), whereas values closer to 1 will increase the contribution of future rewards to the sum.

The value for a given policy π, expressed by Equation 1, can be iteratively computed by the so-called *Bellman Equation* [1]. In value iteration approaches, one starts with arbitrarily chosen value function V_0^{π} and, during each iteration t, for each state $s \in S$ the value function is updated based on the immediate reward and the current estimates of V^{π}:

$$V_{t+1}^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} T(s,\pi(s),s') V_t^{\pi}(s') \tag{2}$$

The process of updating state value functions based on current estimates of successor state values is referred to as *bootstrapping*. The depth of successor states considered in the update can be varied, i.e., one can perform a shallow update where one only looks at immediate successor states or a deep update where successors of successors are also considered. The value function estimates of successor states are used to update the value function estimate of the current state. This is called a *backup* operation. Different algorithms use different backup strategies, e.g., sample backups (sample a single successor state) or full backups (sample all successor states).

The goal of an MDP is to find the optimal policy, i.e., the policy that receives the most reward. The optimal policy $\pi^*(s)$ is such that $V^{\pi^*}(s) \geq V^{\pi}(s)$ for all $s \in S$ and all policies π. It is proven that the optimal value function, often abbreviated as V^*, satisfies the following Bellman optimality criterion:

$$V^*(s) = R(s) + \gamma \max_{\alpha \in A} \left[\sum_{s' \in S} T(s,a,s') V^*(s')\right] \tag{3}$$

Solving Equation 3 can be done in an iterative manner, similar to the computation of the value function for a given policy such as expressed in Equation 2. The Bellman optimality criterion is turned into an update rule:

$$V_{t+1}^{\pi}(s) = R(s) + \gamma \max_{\alpha \in A} \left[\sum_{s' \in S} T(s,a,s') V_n^{\pi}(s')\right] \tag{4}$$

The optimal action can then be selected as follows:

$$\pi^*(s) = \arg\max_a \left[R(s) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \right] \tag{5}$$

Besides learning state-values, one can also define state-action value functions, or so-called *Q-functions*. Q-functions map state-action pairs to values, $Q : S \times A \rightarrow \mathbb{R}$. They reflect the long term desirability of performing action a in state s, and then performing policy π thereafter. The Q-function is defined as follows:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \tag{6}$$

The optimal policy π^* selects the action which maximizes the optimal action value function $Q^*(s,a)$ for each state $s \in S$:

$$\pi^*(s) = \arg\max_a Q^*(s,a) \tag{7}$$

Solutions for and properties of state value functions can be straightforwardly translated to state-action values.

When an environment's model (i.e., transition T function and reward R function) is known, the optimal policy can be computed using a dynamic programming approach, as for example with *policy iteration* or *value iteration*. When the model of the environment is unknown, as it usually is, we can use sampling based techniques such as *temporal difference learning* as an alternative. These techniques do not depend on a model but rather collect samples of interactions with the environment and update the value estimates based on these interactions. An important issue is the used sampling strategy, better known in RL literature as the exploration-exploitation dilemma, i.e., when to explore the environment and when to exploit acquired knowledge. Various exploration and exploitation strategies exist, such as ε-greedy and Boltzmann exploration. For a thorough overview, we refer to [30]. Temporal difference learning methods such as Q-learning [29] and SARSA [20] are model-free solution methods. The algorithms are described in detail in [23]. The update rule for the most popular algorithm, *one-step* Q-learning is denoted as:

$$Q(a,s) \rightarrow (1-\alpha) Q(a,s) + \alpha \left[r + \gamma \max_{a'} Q(a',s') \right] \tag{8}$$

where α is the step-size parameter, and γ the discount-rate. Both algorithms are proven to converge to optimal policies under loose conditions, given that no abstractions have been applied. Unfortunately for complex, real-world problems, solving MDPs is impractical and complexity must be reduced in order to keep learning tractable. In the following section we will discuss the benefit of relational reinforcement learning for generalizing over the state, action, and policy space.

3 Relational reinforcement learning

Relational reinforcement learning (RRL) combines the RL setting with relational learning or inductive logic programming (ILP) [14] in order to represent states, actions, and policies using the structures and relations that identify them. These structural representations allow abstraction from and generalization over specific goals, states, and actions. Because RRL algorithms try to solve problems at an abstract level, the solutions will often carry to different instantiations of that abstract problem. For example, resulting policies learned by an RRL system often generalize over domains with varying number of existing objects.

A straightforward example is the blocks world. A number of blocks with different properties (size, color, ...) is placed on each other or on the floor. It is assumed that an infinite number of blocks can be put on the floor and that all blocks are neatly stacked onto each other, e.g., a block can only be on one other block at the same time. The possible actions consist of moving one clear block (e.g., a block with no other block on top of it) onto another clear block or onto the floor. It is impossible to represent such blocks world states with a propositional representation without an explosion of the number of states. Consider as an example the right-most state in Figure 1. In First-Order Logic (FOL), this state can be represented, presuming this state is called *s*, by the conjunction

$$\{on(s,c,floor) \wedge clear(s,c) \wedge on(s,d,floor) \wedge on(s,b,d) \wedge on(s,a,b) \wedge clear(s,a) \wedge on(s,e,floor) \wedge clear(s,e)\},$$

which is obtained through executing the move-action (indicated by the arrow), noted as *move(r,s,a,b)*, in the previous state *r*.

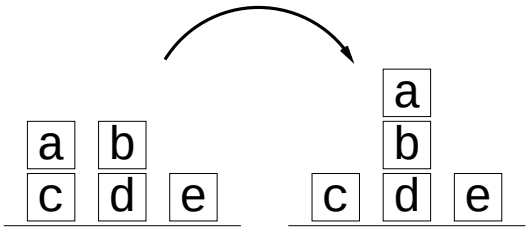

Fig. 1 The blocks world

One of the most important benefits of the relational learning approach is that one can generalize over states, actions, objects selectively. One can abstract away only those parts of the state-space that are less important. As an example, in the blocks world scenario, one can say "there exists a small block which is on a large block" $(\exists B1,B2 : block(B1,small,\_),block(B2,large,\_),on(B1,B2))$. Objects $B1$ and $B2$ are free variables, and can be instantiated by any block in the environment. Hence, RRL can generalize over blocks, and learn policies for a variable number of objects without causing an explosion of the number of states.

Algorithm 1 The Relational Reinforcement Learning Algorithm

Input: initialize $Q(s,a)$ and s_0 arbitrarily
Input: $e \leftarrow 0$
repeat
  $Examples \leftarrow \oslash$
  $i \leftarrow 0$
  repeat
    take a for s using policy $\pi(s)$ and observe r and s'
    $Q(a,s) \rightarrow (1-\alpha)Q(a,s) + \alpha\left[r + \gamma \max_{a'} Q(a',s')\right]$
    $Examples \leftarrow Examples \cup \{a,s,Q(s,a)\}$
    $i \leftarrow i+1$
  until (s_i is terminal)
  Update $\hat{Q}_e$ using *Examples* and a relational regression algorithm to produce $\hat{Q}_{e+1}$
  $e \leftarrow e+1$
until (no more episode)

Although, RRL is a relatively new domain, several approaches have been proposed during the last few years. One of the first methods developed within RRL, was relational Q-learning [6]. A high level description of the algorithm is given in Table 1. Relational Q-learning behaves similar to standard Q-learning, with the exception of the agent's representation of the states S, actions A and learned value-function. In relational reinforcement learning, this representation contains structural or relational information about the environment. Furthermore, it employs a relational regression algorithm to generalize over the value function (and policy) space. Learning examples, stored as a tuple $< a,s,Q(s,a) >$ are processed by an incremental relational regression algorithm to produce a relational value-function (or policy) as a result.[1] So far, a number of different relational regression learners have been developed.[2]

In the current work, we employ the incremental first-order regression tree algorithm $RRL-TG$ [5] as our chosen regression algorithm. The algorithm is denoted in Table 2. Given a set of examples, the algorithm incrementally builds up a tree (*using top down induction*). It starts with an empty tree with one leaf, which contains all stored examples. Then, when the algorithm reaches a leaf node, candidate tests defined in the so-called *language bias* are evaluated. The tree may be expanded by candidate tests that reduce variance among Q-values sufficiently. The best among these candidate tests is then selected to expand the tree.

4 Multi-agent relational reinforcement learning

During the 1990's multi-agent systems (MAS) have become a popular approach in solving computational problems of distributed nature as for instance load balancing

[1] An example of such a relational Q-function is illustrated in Figure 3.

[2] A thorough discussion and comparison can be found in [4].

Algorithm 2 TG Algorithm

Input: input is a collection of examples $< a,s,Q(s,a) >$
Input: initialize tree with single leaf with empty statistics
$i \leftarrow 0$
repeat
sort down $Example_i$ in tree until it reaches leaf and update statistics
if statistics in leaf indicate new split **then**
generate new internal node using indicated test and grow 2 new leafs with empty statistics
end if
$i \leftarrow i+1$
until (no more examples)

or distributed planning systems. They are a conceptually proved solution method for problems of this nature. However, designing a cooperative MAS with both a global high reward for the system and high individual rewards for the different agents is still a difficult problem, see e.g., [9, 17, 18, 26]. The general goal is therefore to have a good system performance where all individual agents in the MAS contribute to some part of the collective through their individual actions.

Several approaches have been proposed, ranging from joint action learners [10] to individual local Q-learners. All of these approaches have their own merits as drawbacks in a multi-agent context. In the first approach, i.e., the joint action space approach, the state and action space are defined as the Cartesian product of the agent's individual state and action spaces. This implies that the state information is shared amongst the agents and actions are taken and evaluated synchronously. It is obvious that this approach leads to very big state-action spaces, and assumes instant communication between the agents. Clearly this approach is in contrast with the basic principles of many contemporary multi-agent applications such as distributed control, asynchronous actions, incomplete information, cost of communication, etc. In the local or selfish Q-learners setting, the presence of the other agents is totally neglected, and agents are considered to be selfish reinforcement learners. The effects caused by the other agents also acting in that same environment are considered as noise. In between these approaches we can find examples which try to overcome the drawbacks of the joint action approach [2, 13, 15, 21, 22, 27]. There has also been quite some effort to extend these RL techniques to Partially Observable MDPs and non-Markovian settings [11].

The complexity of designing a MAS and modeling the behavior of agents within, stems from three important sources. First, partial observability is all but unavoidable in MAS because even if the environment is fully observable, the intentions and plans of other agents cannot be assumed to be known, especially when multiple agents in the MAS are updating their behavior through learning. Local decision making, and therefore inherently partial observability, is a key requirement to make MAS expandable both in the number of agents and in the size of the environment they work in. Agents can more naturally generalize over unseen parts of the environment using a rich relational language. Second, agents in a MAS can interfere with each other's actions, plans or goals. This is yet another reason why joint-action learners

and individual Q-learners mentioned previously are not appropriate for MAS. One needs to take external influences into account when optimizing locally. The problem then translates into learning which other agents to keep in mind when optimizing one's own behavior. Relational representations can be of great help when trying to define (and eventually learn) important relations between agents in MAS, for example in learning with whom to communicate.

Communication is major component of MAS, which can potentially overcome the previously mentioned complexities. Namely, through communication agents can limit the influence of partial observability, and obtain valuable information about other agents. Harmful interactions with agents can be avoided, while on the opposite, agents with common interest can work together to achieve a higher reward. However, communication in MAS is often assumed to be either *limited*, *unreliable*, or *costly*. Consider, for example, the situation where agents can only communicate with agents that are located within a certain range or the cost could be dependent on the distance between agents. Hence, deciding when to communicate with whom adds a layer of complexity to the behavior of agents in a MAS. Therefore, we believe that Multi-Agent Relational Reinforcement Learning (MARRL) applied to the problem of efficient communication can be a valuable contribution to creating scalable, complex MAS.

So far, all work on RRL has focused on the single agent case. To our knowledge, there is almost no existing work on applying RRL in a MAS. Earlier, van Otterlo et al. [16] already mentioned the possible benefits of using RRL in MAS. They state that cognitive and sapient agents especially need a learning component where the RL paradigm is the most logical choice for this. Since these agents are (usually) logic-based, RRL is indicated as a very suitable learning method for intelligent agents. Letia and Precup [12] present a system where multiple agents, represented by GOLOG programs, act in the same environment. The GOLOG programs can supply initial plans and prior knowledge. The agents however can not communicate. There also exist a game-theoretic extension of GOLOG that can be used to implement different agents and compute Nash policy pairs [7]. Hernández et al. [8] use logical decision trees to add a learning factor to BDI (Belief, Desire, Intension) agents. Only Croonenborghs et al. [3] present an approach of using RRL in MAS. They show that when using a relational learner, agents can induce knowledge by observing other agents. However they do not consider the issue of communication between agents. As we consider communication to be one of the crucial aspects of MARRL, we will focus in this paper on using RRL to learn with whom to communicate. In the next Section we describe experiments in an cooperative MAS wherein agents learn an communication policy under several conditions.

5 Empirical evaluation

In this section, we will empirically investigate the benefits of using RRL for communication between agents in an abstract cooperative MAS. More specifically, we

will investigate the influence on the learning performance when allowing agents to seek advice, both in a relational and propositional system. Furthermore, we will vary the quality of the prior knowledge of agents (i.e., agents may not always have the right answer), limit the communication capabilities of agents and finally vary the number of agents and task difficulty in our system. We will also apply relational communication to a challenging multi-state task.

Since we would like to study the effects and benefits of efficient communication, we do not consider the aspects of agent action interference and partial observability in our test environment. Hence, all agents are working on their own task without the possibility to interfere with each other. Not only is this still a challenging problem, an isolated study of the relations between the agents and methods that gain maximally from this, are necessary to handle more complex problems.

5.1 Learning task

The learning task of the agents consists of learning how to complete a number of different tasks. Tasks and agents can be characterized by a number of properties (e.g., color, size etc.) . Every agent is assigned to a random task in the beginning of an episode. An agent receives a positive reward if it completes his task and the episode is ended when all agents completed their tasks. The number of possible actions (henceforth called *primitive actions*) an agent can execute for solving a certain task can vary from task to task. To complete a task the agent has to execute the *optimal* action, a single action from the possible primitive actions for that task.

Some agents have a priori knowledge about certain tasks, these agents are called *experts* for that task. A *pure expert* only has access to the task's optimal action, and therefore has no need for exploring the action space for this particular task. Besides pure experts we also introduce different levels of expertise: agents may have some knowledge about the task at hand, i.e., they know that the optimal action is in some subset of all primitive actions for that task and consequently need to explore a constrained part of the action space. Non-experts must learn a policy to complete the task. Determining if an agent is expert for a task can be done in many ways. For example, we can define an agent as expert if a certain property, or all, match for both task and agent. Seeking advice from an expert immediately completes the task, since its action space only contains the optimal action. However, it may also prove useful seeking advice from an agent that has a high degree of expertise for a task, since it can be expected that this agent learns to complete the task faster.

To evaluate whether communication can improve learning performance, we will run a series of experiments with and without communication. For the first, agents can seek advice from other agents by asking them which action they would execute according to their current policy in the current task of the inquirer. Therefore, the action space of an agent for some task consists of the number of the primitive actions for this task and the *advice actions*, one for every agent in the environment.

5.2 Experimental results

All agents are relational *Q*-learners that use the incremental relational tree learner RRL-TG [5] to learn a generalized *Q*-function (we refer to Section 3 for details). To guide this tree building a declarative bias is specified. This declarative bias contains a language bias to specify the predicates that can be used as tests to partition the tree. For all agents this language bias contains predicates that identify the task and the action taken by the agent. Furthermore, it contains candidate tests that check certain properties of the task at hand and properties of the other agents. The candidate tests are listed in Table 1.

Candidate Test	Explanation
`primitive_action(A,B)`	primitive action A has id B
`advice_action(A,B)`	advice action A is suggested by agent B
`task_has_id(A,B)`	task A has id B
`task_has_property(A,B)`	task A has a certain property B
`agent_has_property(A,B)`	agent A has a certain property B

Table 1 The candidate tests collected in the language bias for building up the tree.

We use two versions for the language bias: one to mimic a propositional learner (i.e., standard Q-learning) and a relational variant. The propositional variant straightforwardly uses the first three candidate tests, wherein agents try out every instantiation of primitive and advice actions until they find the optimal one (e.g., the one that produces a reward of 1). The relational variant of the language bias includes all tests, thereby allowing agents to generalize over properties for the agents and tasks. Given these properties, relational learning agent might discover for a certain task it is smart to seek advice from an agent with certain properties.

During learning, the agents follow a Boltzmann exploration policy [24]. Each agent is maximally allowed 100 actions per episode. Each agent receives a reward of 1.0 if he completes the task and neither reward nor punishment otherwise. Every 100 episodes, the learned policies are tested by freezing the *Q*-function approximation following a greedy policy on 100 test-episodes to check for optimality. Unless mentioned otherwise, all results are averaged over 5 runs. We assume that the transition function is fully deterministic, and that agents have full observability (e.g., agents can see all the other agents, and their properties). Although this is a restricted setting in terms of RL, this is not important for our learning task since we are investigating the effect of communication under several conditions.

We want to investigate whether RRL can generalize over agent and task properties, and therefore can quickly discover experts in the population (if any), so they can communicate (i.e, seek advice) with these experts. We expect this can potentially accelerate the learning process. To validate this claim empirically, we compare the average learning performance for agents in two situations, namely (1) learning with and (2) learning without means for communication. The learning task without

advice boils down to selecting the optimal *primitive action* for a task. If we add *advice actions* to the action space, agents do not necessary need to learn to locate the optimal *primitive action* for each task, it may be much better to seek advice from an expert. This is especially true in our experimental setup, since we do not yet consider communication costs or constraints (although this is certainly possible in our task, and we will elaborate on this in our future work).

5.2.1 Seeking advice

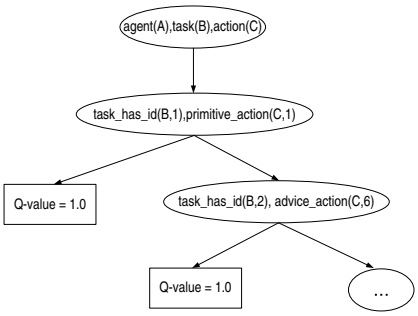

Fig. 2 Example policy learned using the propositional language bias

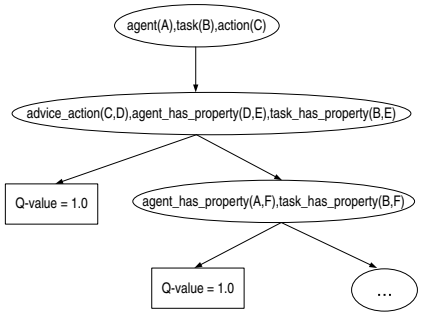

Fig. 3 Example policy learned using the relational language bias

Consider the following experimental setting: we use a MAS containing 10 agents and 10 different tasks. Both agents and tasks are described by one property with a number of possible values (e.g., the property is shape and possible values are circle, square etc.). The tasks all have a different difficulty. In the first task their are only 5 primitive actions available, for the second task this is doubled to 10 primitive actions and so forth until 2560 actions for the tenth task. In this setting, each agent is an expert for a certain task if he shares the property with that task. The experiment is set up in such a way that every agent is expert for exactly one different task and hence there is exactly one expert for every task.

Figure 4 presents the result for the basic setting. It clearly shows the gain that agents with the capability of communication have over agents without this capability. Learning performance is increased by communication, both using a propositional and relational language bias. An example policy learned with the propositional language bias is showed in Figure 2. As we can see, this approach mimics a standard Q-learner where for each instantiation of a task, the optimal action must be located (albeit through personal exploration or by asking a specific agent for advice). No benefit is taking from the fact that certain agents (with a matching property with the task) are experts for solving this particular task. Figure 3 illustrates the learned policy using the relational language bias. This learned policy concisely and elegantly illustrates the optimal policy: for a particular task one should seek advice

from its corresponding expert, otherwise the agent itself is expert. From Figure 4 we may conclude that the generalization of relational learning is essential to take optimal advantage of the added communication.

5.2.2 Influence of restricted communication

We also investigate the influence of restricted communication on the performance of the agents. Figure 5 shows the result when agents can no longer communicate with every other agent. The percentages indicated in the labels reflect the percentage of possible advice actions at every step. The setting *no advice-actions* corresponds to the setting *without communication* in the previous experiments. In the setting *pure experts*, agents may communicate with all other agents. The uncertainty of communication causes a drop in performance, but due to the generalization in the agents' policy, the possibility of communication still has a beneficial effect.

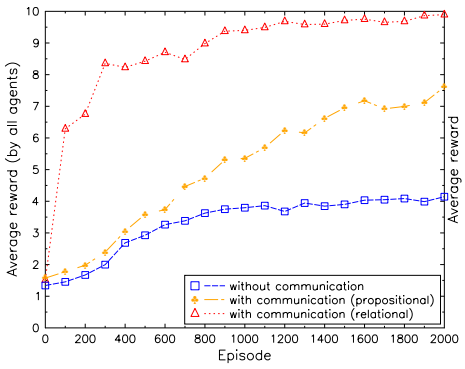

Fig. 4 Communication vs No Communication

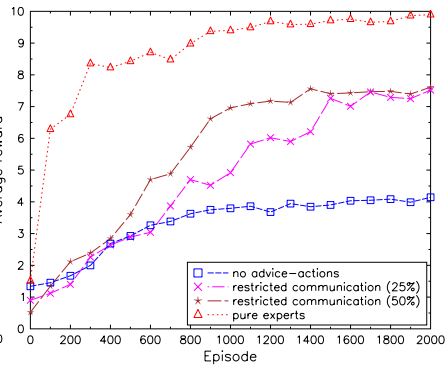

Fig. 5 Influence of restricted communication

5.2.3 Influence of prior knowledge

In these experiments, we want to evaluate the influence of the quality of prior knowledge. To investigate this, we extended the possible actions the agents can execute in a task when they are expert for that task. Figure 6 shows the influence of the prior knowledge that agents have[3]. As previously explained, in the setting *pure experts* the action space of an expert only includes the optimal action. As a result experts always give optimal advice. This is clearly the most ideal setting in terms of agent communication. In the setting *optimal + advice*, the experts can also execute advice actions and hence they still have to learn to select the primitive action instead of asking for advice. In the worst possible setting *no experts*, the agents have no

[3] The results of the pure experts and the agents without communication are repeated for clarity

prior knowledge, i.e., the possible actions for the agents are all primitive actions and advice actions. We also investigated two intermediate settings where a subset of all primitive actions is selected, namely 50% and 75% of the possible primitive actions. We may conclude from the graph that when using relational learning communication, learning performance is improved even when prior knowledge is limited. In the early stages of learning, the extra possibility of seeking advice does not hurt the agent, only in the very beginning the performance is slightly worse due to the bigger action space. Once some agents learned to complete some tasks, this knowledge is distributed through the population resulting in a performance gain.

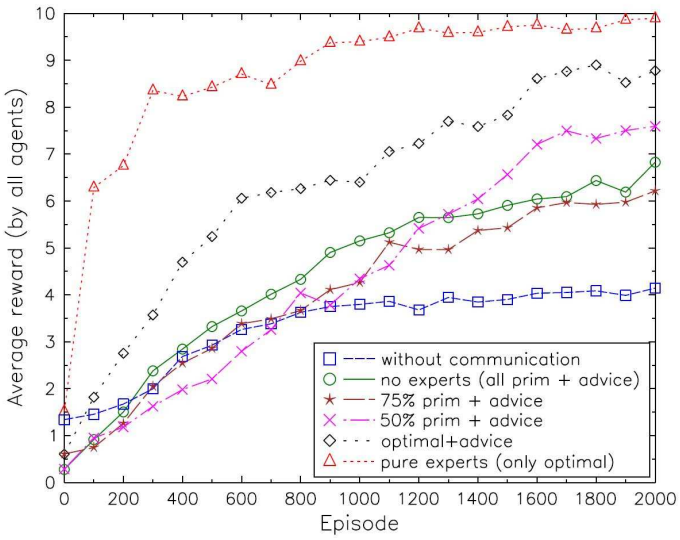

Fig. 6 The influence of prior knowledge

5.2.4 Influence of task difficulty and number of agents

To determine the influence of the task difficulty and the number of agents, we sampled tasks with varying difficulty (i.e., varying the size of action space) restricted by a maximum value. Furthermore, agent and task property values are now randomly sampled and unlike the previous experiments we lose the guarantee that an expert is present in the population. We performed experiments with an increasing maximum value for the action space (this shifts the balance between advice and primitive actions) and with an increasing number of agents compared to tasks (this way we increase the likelihood of finding an exert in the population, but also increases the number of advice actions).

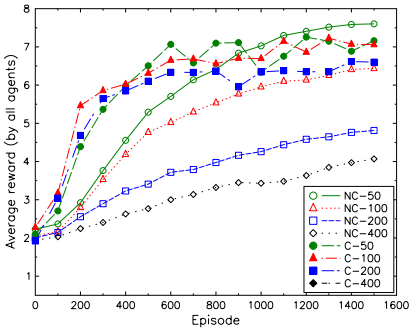

Fig. 7 Influence of the task difficulty

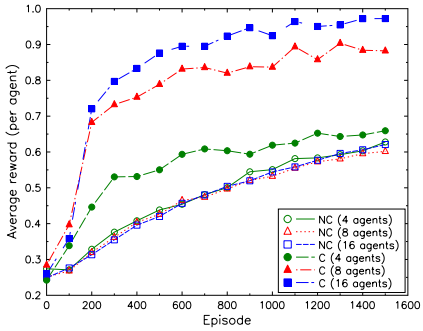

Fig. 8 Influence of the number of agents

In Figure 7 we see that for learning without communication (the experiments denoted with 'NC', the number following this represents the maximum number of actions for this task), agents are steadily moving towards an optimal policy as the number of episodes increases. The convergence rate is determined by the difficulty of the randomly generated tasks. We can also see that for the experiments with advice (denoted with 'C'), agents perform better, in particular in situations where agents are faced with more complex tasks. It seems that when tasks are relatively easy (i.e., see 'NC-50' and 'C50' in Figure 7), it can be at least as efficient to learn to complete the task individually, rather than seeking advice. Only when generating complex tasks, does communication significantly improve learning performance.

Figure 8 shows the results for different sizes of the agent population using the same setting as in the previous paragraph with the maximal number of actions set to 200. The number of agents is denoted by the number behind 'NC' (no communication) or 'C' (communication). The plot shows that the number of agents is irrelevant to the learning performance of the setting without communication. This was expected since agents are learning individually and can not benefit from the knowledge of others. When we allow agents to seek advice, we see much faster learning in particular when many agents are present in the system. A likely explanation is that more agents increases the likelihood of an expert in the population, no matter what task is randomly generated.

5.2.5 Application to complex learning task

In the last series of experiments, we will show that the earlier discussed results also apply for more difficult environments. We extended the number of properties: we provide both agent and task with 3 properties, with respectively 4, 3 and 2 possible values. Therefore, a total of 24 different agent types and task types can be assembled. More precisely, agents and tasks all have a color (with values green, red,

yellow or blue), shape (square, cube or sphere) and size (small or large). The property values for tasks have a weight value associated with them that determine the difficulty of a task. For example, a blue task is 4 times more difficult compared to a yellow task (i.e., has 4 times more possible actions). We vary the task difficulty by setting a minimum and maximum number of allowed actions.

Additionally, we limit both the quality of prior knowledge of agents and move to a multi-state task. Now tasks have to be completed by sequentially solving a number of subtasks (an analogy can be made with an agent moving through a 1-dimensional grid-world to reach a certain goal location). When all subtasks are successfully completed, the agent is rewarded with a reward of 1.0. The latter can be seen as a first step towards multi-state dispersion and cooperation games and is a similar environment as the one used in [3]. We sample random tasks and agents out of the 24 types uniformly. Whereas in the previous settings, pure experts were defined by a single property, now all 3 properties are taken into account. A pure expert (i.e., an agent that can immediately complete the task) must have all properties matched with those of the task. If only a subset of the properties match, the agent will have a certain degree of expertise, depending on the number of matching properties and their type (e.g., a matching color property may have a larger influence than a matching shape property). Given a set of agent and task properties, a degree of expertise can be calculated. If an agent is, let us say 80% expert of a task, the agent only needs to explore 20% of the action space. Of course, if no properties match, the agent has no expertise at all and must explore the whole action space for this task. Again, in this setting it is possible that no, or very few experts are generated for the tasks at hand.

Figure 9 presents the result obtained for a multi-state task consisting of four subtasks. It can be noted that when using RRL, communication clearly helps the agent to learn a better policy faster.

6 Conclusions

In this paper we introduced the novel idea of cross-fertilization between relational reinforcement learning and multi-agent systems to perform well complex multi-state dynamic planning tasks. We proposed to use a relational representation of the state space in multi-agent reinforcement learning as this has many proved benefits over the propositional one, as for instance handling large state spaces, a rich relational language, modeling of other agents without a computational explosion, and generalization over new derived knowledge.

We started this paper with a short introduction to relational reinforcement learning and multi-agent learning. Furthermore, we investigated the positive effects of relational reinforcement learning applied to the problem of agent communication in MAS. More precisely, we investigated the learning performance of RRL given some communication constraints. Based on our empirical results in our abstract cooperative MAS we make the following conclusions:

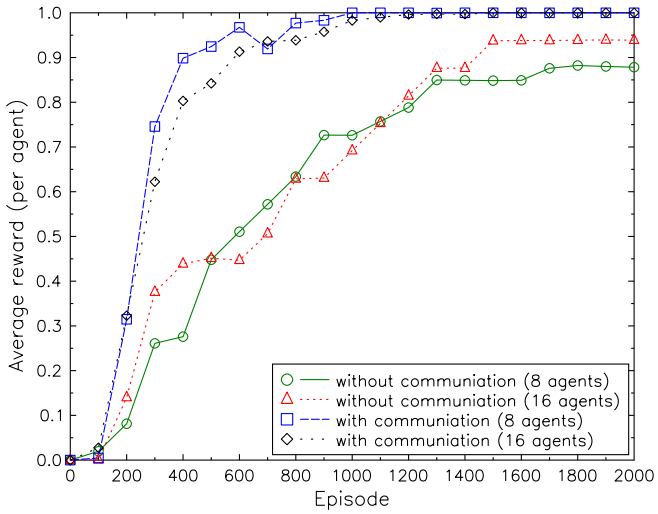

Fig. 9 Multi-state task

1. Communication in general pays off in MAS, and in particular when employing a relational representation of the MAS (see Figure 4).
2. Communication is still beneficial to the learning performance, even when communication is limited (see Figure 5) or unreliable (see Figure 6).
3. Communication is in particular useful for MAS with large number of agents (see Figure 8), and complex tasks (see Figure 7). Therefore, our approach scales well to more complex MAS.
4. Rapid convergence is achieved in more complex multi-state problem (see Figure 9).

In our future work we plan to continue along this track and gradually extend our approach to even more complex and realistic settings e.g. including more elaborate communication possibilities, a cost model, and interference between agents since this is an important part of every multi-agent system. Furthermore, a thorough theoretical study of the described properties and settings will be part of our future research.

References

1. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton, New Jersey (1957)

2. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multi-agent systems. In: Proceedings of the 15th International Conference on Artificial Intelligence, p.746-752 (1998)
3. Croonenborghs, T., Tuyls, K., Ramon, J., Bruynooghe, M.: Multi-agent relational reinforcement learning. Explorations in multi-state coordination tasks. In: Learning and Adaptation in Multi Agent Systems: First International Workshop , LAMAS 2005, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 3898, pp. 192–206. Springer Berlin / Heidelberg (2006). URL = http://www.cs.kuleuven.ac.be/cgi-bin-dtai/publ_info.pl?id=41977
4. Driessens, K.: Relational reinforcement learning. Ph.D. thesis, Department of Computer Science, Katholieke Universiteit Leuven (2004). `http://www.cs.kuleuven.be/publicaties/doctoraten/cw/CW2004\_05.abs.html`
5. Driessens, K., Ramon, J., Blockeel, H.: Speeding up relational reinforcement learning through the use of an incremental first order decision tree learner. In: L. De Raedt, P. Flach (eds.) Proceedings of the 12th European Conference on Machine Learning, *Lecture Notes in Artificial Intelligence*, vol. 2167, pp. 97–108. Springer-Verlag (2001)
6. Džeroski, S., De Raedt, L., Driessens, K.: Relational reinforcement learning. Machine Learning **43**, 7–52 (2001)
7. Finzi, A., Lukasiewicz, T.: Game theoretic golog under partial observability. In: AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, pp. 1301–1302. ACM, New York, NY, USA (2005). DOI http://doi.acm.org/10.1145/1082473.1082743
8. Guerra-Hernández, A., Fallah-Seghrouchni, A.E., Soldano, H.: Learning in BDI multi-agent systems. Lecture Notes in Computer Science **3259**, 218–233 (2004)
9. Hoen, P., Tuyls, K.: Engineering multi-agent reinforcement learning using evolutionary dynamics. In: Proceedings of the 15th European Conference on Machine Learning (2004)
10. Hu, J., Wellman, M.P.: Experimental results on Q-learning for general-sum stochastic games. In: ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 407–414. Morgan Kaufmann Publishers Inc. (2000)
11. Kaelbling, L., Littman, M., Moore, A.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research (1996)
12. Letia, I.A., Precup, D.: Developing collaborative golog agents by reinforcement learning. International Journal on Artificial Intelligence Tools **11**(2), 233–246 (2002)
13. Littman, M.: Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the Eleventh International Conference on Machine Learning, p 157 - 163 (1994)
14. Muggleton, S., De Raedt, L.: Inductive logic programming : Theory and methods. Journal of Logic Programming **19,20**, 629–679 (1994)
15. Nowé, A., Parent, J., Verbeeck, K.: Social agents playing a periodical policy. In: Proceedings of the 12th European Conference on Machine Learning, p 382 - 393, Freiburg (2001)
16. van Otterlo, M.: A characterization of sapient agents. In: International Conference Integration of Knowledge Intensive Multi-Agent Systems (KIMAS-03). Boston, Massachusetts (2003)
17. Panait, L., Luke, S.: Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems **11**(3), 387–434 (2005)
18. Panait, L., Tuyls, K., Luke, S.: Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. Journal of Machine Learning Research **9**(Mar), 423–457 (2008)
19. Puterman, M.: Markov decision processes: Discrete stochastic dynamic programming. John Wiley and Sons, New York (1994)
20. Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. Tech. rep., Cambridge University Engineering Department (1994)
21. Sen, S., Airiau, S., Mukherjee, R.: Towards a Pareto-optimal solution in general-sum games. In: in the Proceedings of the Second Intenational Joint Conference on Autonomous Agents and Multiagent Systems, (pages 153-160), Melbourne, Australia, July 2003 (2003)
22. Stone, P.: Layered learning in multi-agent systems. Cambridge, MA: MIT Press (2000)
23. Sutton, R., Barto, A.: Reinforcement Learning: an introduction. MIT Press, Cambridge, MA, USA (1998)

24. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA (1998)
25. Tadepalli, P., Givan, R., Driessens, K.: Relational reinforcement learning: An overview. In: Proceedings of the ICML'04 Workshop on Relational Reinforcement Learning (2004)
26. Tumer, K., Wolpert, D.: COllective INtelligence and Braess' Paradox. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence, pages 104-109. (2000)
27. Tuyls, K., Verbeeck, K., Lenaerts, T.: A selection-mutation model for Q-learning in Multi-Agent Systems. In: The second International Joint Conference on Autonomous Agents and Multi-Agent Systems. ACM Press, Melbourne, Australia (2003)
28. van Otterlo, M.: The logic of adaptive behavior: Knowledge representation and algorithms for the Markov decision process framework in first-order domains. Ph.D. thesis, Department of Computer Science, University of Twente, Enschede, The Netherlands (2008). May, 512pp
29. Watkins, C.: Learning with delayed rewards. Ph.D. thesis, Cambridge University (1989)
30. Wiering, M.: Explorations in efficient reinforcement learning. Ph.D. thesis, Universiteit van Amsterdam (1999)